UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

TWO EXACT ALGORITHMS FOR THE DISTANCE CONSTRAINED VEHICLE ROUTING PROBLEM

by

Gilbert Laporte*, Yves Nobert**
and Martin Desrochers*

* Ecole des Hautes Etudes Commerciales de Montréal
5255, avenue Decelles, Montréal, Québec, Canada H3T 1V6

** Département des Sciences Administratives, Université du Québec
à Montréal, Montréal, Québec, Canada H3C 3P8

The authors wish to thank the Canadian Natural Sciences and Engineering Research Council (grant A4747) and the Québec Government (FCAC program) for their financial support. Thanks are also due to Ann Laporte for her helpful advice.

Université de Montréal

Centre de recherche sur les transports - Publication #259
Ecole des Hautes Etudes Commerciale - Rapport de recherche #G-82-05

Mai 1982

Dépôt légal - 3e trimestre 1982

ABSTRACT

This paper considers a version of the vehicle routing problem in which all vehicles are identical and where the distance travelled by any vehicle may not exceed a prespecified upper bound. The problem is first formulated as an integer program which is solved by means of a constraint relaxation procedure. Two exact algorithms are developed: one based on Gomory cutting planes and one on branch and bound. Numerical results are reported.

1. INTRODUCTION

In the Operational Research literature, the "Vehicle Routing Problem" (VRP) refers to a family of problems in which a fleet of m vehicles (possibly of different types) must visit at least cost a set of n "cities" or points. (See [1,2,5,16] for excellent surveys of the VRP.) It is assumed that all trips originate and end at the same city or depot (city 1, say) and that each of the remaining n-1 cities must be visited once and only once by a vehicle. In this paper, the matrix C of distances c_{ij} between each pair of cities i and j is symmetrical. C is said to be Euclidean if and only if $c_{ij} \leq c_{ik} + c_{kj}$ for all i,j,k.

The need to resort to many vehicles usually stems from various constraints which arise in practice. Two common types of constraints are:

(i) capacity constraints: the goods delivered (or collected) at city i have a weight (or a volume) equal to d_i. The capacity (defined in terms of weight or volume) of any vehicle must not be exceeded;

(ii) distance constraints: the length of any trip (defined in terms of mileage or time) must not exceed a prespecified limit L. The distinction between mileage and time is irrelevant if all vehicles in the fleet are assumed to travel at the same constant speed.

Many cases of the VRP treated in the literature [3-8,10-16, 26,27] include a combination of constraints (i) and (ii). If these constraints are relaxed, the VRP reduces to a travelling salesman problem (TSP [9]) if $m = 1$ and to a multiple travelling salesman problem (m-TSP [20]) if $m \geq 1$.

The VRP appears to be much more difficult to solve than the TSP or the m-TSP involving the same number of cities. This explains

in part the lack of exact algorithms for the VRP. For example, VRP's with constraints (i) only or (i) and (ii) have been solved optimally for up to about 25-30 cities [6,7]. This compares with 120 cities for the symmetrical TSP [17].

To our knowledge, no exact algorithms have been devised to deal expressly with constraints (ii), although there exist practical applications of the VRP where distance constraints only are present e.g. the case of a sales representative visiting customers but delivering and picking up no goods. We present two such algorithms in this paper.

2. FORMULATION

It is assumed that the cities are to be visited by m identical vehicles. The value of m is either fixed (i.e. determined a priori) or free (i.e. considered as a decision variable). In the latter case, there is no difficulty in principle in imposing a lower bound or an upper bound on m. Let L be the upper bound on the maximum distance travelled by a vehicle. We also define

$$N = \{1,\ldots,n\}$$

$$\bar{S} = N-S \qquad (S \subseteq N)$$

$$H_v(S) \begin{cases} \text{the optimal value of the TSP solution through all points of S (and involving no point outside S) if } v = 1 \\ \text{the optimal value of the v-TSP solution through all points of S (and involving no point outside S) if } 1 \in S \\ \text{undefined otherwise} \end{cases}$$

$V(S)$: a lower bound on the number of vehicles required to visit city 1 and all points of S in the optimal solution

$\lceil t \rceil$: the smallest integer greater than or equal to t.

$$x_{ij} = \begin{cases} 1 & \text{if a vehicle is used to travel directly between } i \text{ and } j \quad (i,j \in N-\{1\}) \\ 2 & \text{if a vehicle is used on a return trip between } i = 1 \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

(N.B. x_{ij} is only defined if $i < j$ and $c_{ij} \leq L$.) The problem can be formulated as follows:

$$(P) \quad \text{minimize} \quad z = \sum c_{ij} x_{ij}$$

subject to

$$(1) \quad \sum_{j=2}^{n} x_{1j} = 2m$$

$$(2) \quad \sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (k \in N-\{1\})$$

$$(3) \quad \sum_{i,j \in S} x_{ij} \leq |S| - V(S) \qquad (S \subseteq N-\{1\},\ |S| \geq 2)$$

$$(4) \quad \sum_{j \in S-\{1\}} x_{1j} + 2 \sum_{\substack{i \in S-\{1\},\ j \in \bar{S} \\ \text{or } i \in \bar{S},\ j \in S-\{1\}}} x_{ij} \geq 3 \qquad (S \subseteq N,\ 1 \in S,\ |S| \geq 2,\ H_1(S) > L)$$

$$(5) \quad x_{1j} = 0,1,2 \qquad (j \in N-\{1\} \text{ and } c_{1j} \leq \tfrac{1}{2}L)$$

$$x_{ij} = 0,1 \qquad \text{otherwise}$$

$$m \geq 1 \text{ and integer}$$

This model possesses a structure similar to those used for the TSP [9,23,24] and for the m-TSP [20]. Constraints (1) and (2) specify the degree of each node and constraints (5) ensure that all variables are integer. Constraints (3) are subtour elimination constraints. They generalize the corresponding constraints (with V(S) = 1) set up by Dantzig et al. for the TSP [9]. Their validity can be established by making the following observation.

Lemma: Let $S \subseteq N-\{1\}$ and $|S| \geq 2$. Assume that the x_{ij}'s satisfy constraints (2) for each $k \in S$. Then inequalities (6) and (7) are equivalent for any real number r:

$$(6) \qquad \sum_{i,j \in S} x_{ij} \leq |S| - r$$

$$(7) \qquad \sum_{\substack{i \in S,\ j \in \bar{S} \\ \text{or } i \in \bar{S},\ j \in S}} x_{ij} \geq 2r$$

Proof: the proof is fairly straightforward. It generalizes the special case where $r = 1$ proved in [9]. ■

Since each vehicle used for visiting a city in S will have to enter and leave S, it follows that constraints (8) must be satisfied in the optimal solution:

$$(8) \qquad \sum_{\substack{i \in S,\ j \in \bar{S} \\ \text{or } i \in \bar{S},\ j \in S}} x_{ij} \geq 2V(S) \qquad (S \subseteq N-\{1\},\ |S| \geq 2)$$

By the lemma, this is equivalent to (3).

Note that constraints (8) could have been used in the formulation instead of constraints (3). The form containing the smaller number of coefficients in the left hand side is in fact always chosen.

Constraints (4) are also subtour elimination constraints. As we shall see later, they are used in non-Euclidean problems to eliminate subtours connected to the depot but involving a set S of cities such that $H_1(S) > L$. Such subtours can be broken by either partitioning S into smaller subtours connected to the depot, in which case, the following inequality will hold true:

$$(9) \qquad \sum_{j \in S-\{1\}} x_{1j} \geq 4$$

or by creating exactly one new positive link between S-{1} and $\bar{S}$, i.e.

$$(10) \qquad \sum_{\substack{i \in S-\{1\},\ j \in \bar{S} \\ \text{or } i \in \bar{S},\ j \in S-\{1\}}} x_{ij} = 1$$

and

$$(11) \qquad \sum_{j \in S-\{1\}} x_{1j} \geq 1$$

or by creating at least two new positive links between S-{1} and $\bar{S}$, i.e.

$$(12) \qquad \sum_{\substack{i \in S-\{1\},\ j \in \bar{S} \\ \text{or } i \in \bar{S},\ j \in S-\{1\}}} x_{ij} \geq 2$$

In all three cases, one constraint of type (4) can be imposed.

3. ALGORITHMS

As for the m-TSP and various other TSP extensions [21,22,25] possessing a similar structure, we have chosen to solve (P) via a constraint relaxation procedure. We present two algorithms based on the two main ways of reaching integrality in (P).

3.1 A cutting planes algorithm

Step 1: A relaxed version of (P) is first solved. This version includes the objective function, the degree constraints and the upper bounds on the x_{ij}'s.

Step 2: A check for integrality is made. If all variables are integer, step 3 is executed. Otherwise, integrality is gradually reached by the introduction of Gomory cutting planes [18]. One Gomory cut is introduced for each fractional solution and the problem is reoptimized. The choice of the best cut is made according to the depth criterion used in [24]. Cuts are deleted from the program as soon as they become ineffective. Experience has shown that the probability of generating the same cut twice is very small.

Step 3: At this point, the solution consists of a number of subtours. A subtour defined on $S \subsetneq N$ is a Hamiltonian cycle through all points of S and involving no point outside S. To each subtour, we associate a real number $\ell(S)$ equal to the current value of $\sum_{i,j \in S} c_{ij} x_{ij}$.

The program then proceeds to check whether the solution contains any illegal subtours. There are two types of illegal subtours:

Type 1: those for which $1 \notin S$;

Type 2: those for which $1 \in S$ and $H_1(S) > L$.
Note that $H_1(S)$ is bounded below by $\ell(S)$ since the relaxed problem is optimal.

If no illegal subtour can be found, the algorithm terminates. Otherwise, for each illegal subtour, the appropriate subtour elimination constraint is generated according to whether C is Euclidean or not and to the type of subtour encountered. The problem is reoptimized.

EUCLIDEAN PROBLEMS

Here, both types of subtours are eliminated by a type (3) constraint. Since the problem is currently optimal, it follows that

(13) $\ell(S) \leq H_1(S) \leq H_1(S \cup \{1\})$

Also, since C is Euclidean, it can easily be verified that

(14) $H_v(S \cup \{1\}) \leq H_v(S' \cup \{1\})$ $\qquad (v=1,\ldots, |S-\{1\}|)$

for all S' such that $S \subset S'$, and that

(15) $H_v(S \cup \{1\}) \leq H_{v+1}(S \cup \{1\})$ $\qquad (v=1,\ldots, |S-\{1\}|-1)$

The minimum distance which will have to be travelled by all vehicles visiting city 1 and all points of S while satisfying all constraints, is therefore bounded below by $\ell(S)$ and by $H_v(S \cup \{1\})$ for some v.

The subtour can be broken by imposing a constraint of type (3), i.e.

(16) $$\sum_{i,j \in S-\{1\}} x_{ij} \leq |S-\{1\}| - V(S-\{1\})$$

where $V(S-\{1\})$ is equal to the smallest value of v satisfying

(17) $v = \lceil H_v(SU\{1\})/L \rceil$ $\qquad (v=\lceil \ell(S)/L \rceil, \ldots, |S-\{1\}|)$

Tests made over 30 problems (generated as in section 4) indicate that in practice, (17) is always satisfied for $v = \lceil \ell(S)/L \rceil$. It is therefore unnecessary to compute successive values of $H_v(SU\{1\})$ since this does not increase v. (If too low a value of v is chosen, the constraint is still valid, but weaker.)

NON-EUCLIDEAN PROBLEMS

Type 1 illegal subtours

Since $1 \notin S$, and because C is non-Euclidean, (14) does not necessarily hold and V(S) is set equal to 1 in constraint (3). We therefore impose the same subtour elimination constraint as in the TSP [9].

Type 2 illegal subtours

Such subtours are eliminated by the imposition of a constraint of type (4). The derivation of these constraints has already been presented in section 2.

After step 3 has been executed, the program proceeds to step 2.

3.2 A branch and bound algorithm

Step 1: Define a first subproblem as in step 1 of the previous algorithm.

Step 2: Choose the next subproblem to be solved, according to the rule described in [18].

Step 3: Solve the subproblem using simplex. Let $\bar{z}$ be the cost of the optimal solution and z^* the cost of the best solution so far identified. If $\bar{z} > z^*$, go to step 7. If at least one variable is fractional, go to step 6.

Step 4: The current solution is integer. A check for illegal subtours (see section 3.1) is made. If the solution contains no illegal subtour, the algorithm proceeds to step 5. Otherwise, a subtour elimination constraint is generated for each illegal subtour as in 3.1. The subproblem is reoptimized in step 3.

Step 5: Set $z^* = \bar{z}$ and store the current solution. Go to sept 7.

Step 6: Identify a variable x_t on which to branch and create two (or three) new subproblems by setting $x_t = 0$ and $x_t = 1$ (and sometimes $x_t = 2$). Go to step 8.

Step 7: The backtracking procedure described in [18] is applied and the next subproblem to be optimized is identified. Let λ be the level of the branch and bound tree associated with this subproblem. All subtour elimination constraints introduced at a level $\lambda' \geq \lambda$ must be eliminated: ineffective constraints are simply removed; effective constraints are first made ineffective by modifying their right-hand side. In constraints (3), the right-hand side is replaced by $|S| + 1$, while in constraints (4), it is replaced by -1. The current solution is then updated (see [19], chapter 7) and the subproblem is reoptimized. (Removing subtour elimination constraints is made necessary by the fact that the bounds $H_1(S)$, $V(S)$ and $\ell(S)$ used for their generation are only valid for a subset of the feasible region of (P), characterized by the restrictions imposed on the variables at levels $\lambda' \geq \lambda$.)

Step 8: The list of candidate subproblems is updated. If this list is empty, the optimal solution has been found and the algorithm terminates. Otherwise, step 2 is executed.

4. COMPUTATIONAL RESULTS

The two algorithms were tested on a number of randomly generated problems ranging from 20 to 60 cities. Euclidean problems were obtained by generating points (x_i, y_i) uniformly in $[0,100]^2$ and by defining c_{ij} as the Euclidean distance between (x_i, y_i) and (x_j, y_j). Cost coefficients for "non-Euclidean" problems were randomly generated from a uniform distribution on [0,100]. All problems were solved on the University of Montreal CYBER 173 computer, using an FTN5 compiler. The two algorithms were constructed by adding subroutines to or by otherwise modifying the Land-Powell MIF and BB codes [18].

In the first set of problems (tables 1 to 4) m was free (i.e. it was considered as a variable). For each value of n, 10 problems were generated (5 Euclidean and 5 non-Euclidean) and solved for different values of L, using both algorithms. Some problems failed for one of the following reasons:

(i) time limit reached (300 seconds, CPU time);

(ii) excessive memory ($200\ 000_8$ words);

(iii) the determinant of the inverse basis exceeded a preset limit of 10^7.

Tables 1 to 4 indicate the number of successful problems out of 5 and the average performance of the successful problems.

Time was the main factor limiting the size of the problems that could be handled by either algorithm. (In the m-TSP, memory appeared to be the limiting factor [20]). In general, the branch and bound algorithm fared worse than the cutting planes algorithm. This is due, to a large extent, to the fact that in the branch and bound algorithm, constraints that have been introduced in a subproblem must be eliminated when backtracking (see step 7). Many simplex iterations are sometimes required before effective subtour elimination constraints can be made ineffective. As has already been observed in studies on the TSP and similar problems [20-24], non-Euclidean problems are much easier to solve than Euclidean problems.

For both algorithms, the value of L was allowed to vary from 1000 downwards and was chosen (after some empirical tests) in such a way that the maximum distance constraints would have a high probability of being binding. As could be expected, problems became more difficult as L decreased. The total number of Gomory cuts (in the first algorithm) and of subtour elimination constraints (in both algorithms) remained relatively low. The column "maximum size of the inverse" represents the maximum size of the reduced basis in the Land-Powell routines [18]: it refers to the maximum number of effective constraints during the course algorithm. Since all degree constraints are effective in most problems, it can be seen that relatively few additional constraints were effective during the course of the algorithm. This explains to a large extent why our constraint relaxation approach is so successful.

Our results indicate that the cutting planes algorithm can handle problems involving 50 cities in the Euclidean case and 60 cities in the non-Euclidean case. In comparison, Christofides [3]

reported in 1976 that, at that time, the largest VRP of any complexity solved by an exact algorithm involved 31 cities. As far as the authors are aware, this figure has not been improved since.

Tests were also carried out to examine the performance of the algorithms in the case where m was a predetermined constant. For various values of n and L, we let m successively take the values 2,4,6,8 and 10. The results were then compared with the corresponding results for m free. Contrary to our expectations, no significant difference was observed between m fixed and m free; furthermore, the difficulty of the problems did not seem to vary according to the size of m. In contrast, earlier work on the m-TSP [20] had indicated that problems where m was fixed were in general easier to solve than when m was free and that larger values of m yielded shorter solution times.

TABLE 1: Euclidean Problems - Cutting Planes Algorithm

n	L	number of successful problems out of 5	number of iterations	maximum size of the inverse	number of Gomory cuts	number of subtour elimination constraints	time (seconds)
20	1000	4	59.0	23.5	1.0	4.5	2.19
	400	4	63.5	25.5	1.5	6.3	3.30
	350	3	78.0	29.3	3.7	9.3	4.86
	325	3	94.0	31.0	6.3	15.3	7.27
	300	3	138.7	32.0	13.7	16.3	13.36
30	1000	5	120.0	39.4	5.8	10.2	11.43
	450	4	121.0	40.3	5.0	11.0	13.88
	400	5	240.6	45.4	23.6	19.2	39.31
	350	5	375.0	46.0	42.6	23.8	62.72
	300	0	-	-	-	-	-
40	1000	5	218.2	53.8	14.8	17.0	49.01
	500	3	335.3	55.3	34.0	24.0	89.10
	450	5	292.0	58.2	18.6	32.4	77.85
	400	5	371.8	59.0	23.0	35.2	104.69
	350	4	423.8	59.8	29.0	35.8	122.39
50	1000	3	278.3	63.7	16.0	24.0	87.17
	650	3	278.3	63.7	16.0	24.0	87.17
	600	3	278.3	63.7	16.0	24.0	87.17
	550	2	465.5	68.0	21.5	28.5	159.93
	500	2	635.0	69.5	43.0	40.5	269.74
60	1000	1	316.0	65.0	13.0	0	109.10
	700	1	316.0	65.0	13.0	0	109.10
	650	1	316.0	65.0	13.0	0	109.10
	600	1	316.0	65.0	13.0	0	109.10
	550	1	316.0	65.0	13.0	0	109.10

TABLE 2: Euclidean Problems - Branch and Bound Algorithm

n	L	number of successful problems out of 5	number of iterations	maximum size of the inverse	number of subtour elimination constraints	time (seconds)
20	1000	4	72.5	22.5	4.8	2.17
	400	4	91.8	24.0	6.5	16.41
	350	4	405.8	27.3	27.0	16.70
	325	4	1842.3	29.3	84.5	84.93
	300	3	788.0	28.7	49.7	35.75
30	1000	3	728.0	35.7	44.3	47.70
	450	2	846.0	37.0	44.0	50.76
	400	0	-	-	-	-
	350	0	-	-	-	-
	300	0	-	-	-	-

TABLE 3: Non-Euclidean Problems - Cutting Planes Algorithm

n	L	number of successful problems out of 5	number of iterations	maximum size of the inverse	number of Gomory cuts	number of subtour elimination constraints	time (seconds)
20	1000	5	50.6	22.0	0.2	2.4	1.69
	200	5	60.8	24.0	2.4	5.2	2.44
	175	4	65.3	25.5	4.5	6.7	2.95
	150	3	89.7	27.0	8.7	10.0	5.42
	125	1	54.0	24.0	2.0	5.0	2.03
30	1000	5	74.4	32.6	1.2	2.8	5.29
	200	5	120.6	36.0	8.8	5.4	14.29
	175	5	171.2	39.0	16.6	8.8	23.95
	150	5	186.6	40.4	21.0	10.8	26.86
	125	3	202.7	40.7	26.7	12.0	30.62
40	1000	5	159.4	46.8	11.0	7.4	30.39
	200	4	191.5	48.8	15.5	10.0	38.70
	175	2	194.0	47.5	13.5	14.0	48.68
	150	1	144.0	45.0	2.0	4.0	18.35
	125	1	202.0	47.0	13.0	10.0	36.53
50	1000	5	161.4	51.6	1.4	2.4	29.06
	200	5	173.4	52.8	3.0	3.6	33.84
	175	4	196.6	55.5	7.5	6.2	46.73
	150	4	223.8	55.5	10.5	8.0	45.80
	125	1	190.0	58.0	11.0	5.0	39.54
60	1000	5	194.4	61.6	2.0	2.4	50.70
	200	5	209.0	62.4	4.2	4.2	58.56
	175	3	251.0	65.0	8.3	8.3	77.44
	150	2	224.0	63.5	5.0	6.5	63.46
	125	2	284.5	66.5	13.5	7.5	107.48

TABLE 4: Non-Euclidean Problems - Branch and Bound Algorithm

n	L	number of successful problems out of 5	number of iterations	maximum size of the inverse	number of subtour elimination constraints	time (seconds)
20	1000	5	53.2	21.8	1.8	1.38
	200	5	84.8	22.8	7.4	2.94
	175	4	157.0	23.8	12.0	6.48
	150	3	410.7	24.0	13.0	16.15
	125	1	72.0	24.0	6.0	2.52
30	1000	5	99.8	31.6	3.8	6.00
	200	5	234.0	33.4	12.2	15.34
	175	5	1064.8	34.4	28.4	72.80
	150	3	459.4	34.3	24.3	36.76
	125	0	-	-	-	-
40	1000	5	664.6	44.4	19.4	83.34
	200	2	560.5	44.0	22.0	70.87
	175	2	1075.5	43.5	25.0	125.87
	150	1	178.0	43.0	4.0	17.33
	125	0	-	-	-	-
50	1000	5	358.2	50.6	12.6	70.92
	200	4	169.0	49.5	3.0	21.48
	175	3	471.3	51.0	12.7	78.62
	150	2	461.5	49.5	4.5	65.48
	125	1	466.0	52.0	10.0	72.32
60	1000	5	328.2	60.6	5.2	70.94
	200	4	349.8	60.8	5.8	74.30
	175	1	156.0	58.0	1.0	25.77
	150	1	156.0	58.0	1.0	25.71
	125	1	188.0	58.0	2.0	48.06

BIBLIOGRAPHY

[1] BODIN, L.D., GOLDEN, B.L.: "Classification in Vehicle Routing and Scheduling", Networks, vol.11, pp.97-108, 1981.

[2] BODIN, L.D., GOLDEN, B.L., ASSAD, A., BALL, M.: "The State of the Art in Routing and Scheduling of Vehicles and Crews", Report No. UMTA/BMGT/MSS #81-001, U.S. Department of Transportation, Washington D.C., 396 pp., 1981.

[3] CHRISTOFIDES, N.: "The Vehicle Routing Problem", RAIRO (recherche opérationnelle), vol.10, pp.55-70, 1976.

[4] CHRISTOFIDES, N., EILON, S.: "An Algorithm for one Vehicle-dispatching Problem", Operational Research Quarterly, vol.20, pp.309-318, 1969.

[5] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "The Vehicle Routing Problem", in Combinatorial Optimization, pp.315-338, Wiley, 1979.

[6] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Mathematical Programming, vol.20, pp.255-282, 1981.

[7] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems", Networks, vol.11, pp.145-164, 1981.

[8] CLARKE, G., WRIGHT, J.W.: "Scheduling of Vehicles from a Central Depots to a Number of Delivery Points", Operations Research, vol.12, pp.568-581, 1964.

[9] DANTZIG, G.B., FULKERSON, R., JOHNSON, S.: "Solution of a Large-Scale Travelling-Salesman Problem", Operations Research, vol.2, pp.393-410, 1954.

[10] DANTZIG, G.B., RAMSER, J.H.: "The Truck Dispatching Problem", Management Science, vol.6, pp.81-91, 1959.

[11] DINKEL, J.J., KLEINDORFER, G.B., KOCHENBERGER, G.A., WONG, S.N.: "Environmental Inspection Routes and the Constrained Travelling System Salesman Problem", Computers and Operations Research, vol.3, pp.269-282, 1976.

[12] FISHER, M., JAIKUMAR, R.: "A Generalized Assignment Heuristic for Vehicle Routing", Networks, vol.11, pp.109-124, 1981.

[13] GASKELL, T.J.: "Bases for Vehicle Fleet Scheduling", Operational Research Quarterly, vol.18, pp.281-295, 1967.

[14] GAVISH, B., SHLIFER, E.: "An Approach for Solving a Class of Transportation Scheduling Problems", European Journal of Operational Research, vol.3, pp.122-134, 1978.

[15] GILLET, B.E., MILLER, L.R.: "A Heuristic Algorithm for the Vehicle-Dispatch Problem", Operations Research, vol.22, pp.340-349, 1974.

[16] GOLDEN, B.L., MAGNANTI, T.L., NGUYEN, H.Q.: "Implementing Vehicle Routing Algorithms", Networks, vol.7, pp.113-148, 1977.

[17] GRÖTSCHEL, M.: "On the Symmetrical Travelling Salesman Problem: Solution of 120-city problem", Mathematical Programming Study, no 12, North-Holland, pp.61-71, 1980.

[18] LAND, A.H., POWELL, S.: "FORTRAN Codes for Mathematical Programming: Linear, Quadratic and Discrete", Wiley, 249 pp., 1973.

[19] LAPORTE, G.: "Permutation Programming: Problems, Methods and Applications", Ph.D. Thesis, University of London, 358 pp., 1975.

[20] LAPORTE, G., NOBERT, Y.: "A Cutting Planes Algorithm for the m-Salesmen Problem", Journal of the Operational Research Society, vol.31, pp.1017-1023, 1980.

[21] LAPORTE, G., NOBERT, Y.: "An Exact Algorithm for Minimizing Routing and Operating Costs in Depot Location", European Journal of Operational Research, vol.6, pp.224-226, 1981.

[22] LAPORTE, G., NOBERT, Y., MERCURE, H.: "The Multi-Depot m-Salesmen Problem", Methods of Operations Research, Oelgeschlager, Gunn & Hain Publishers, Cambridge, Massachusetts, pp.367-370, 1981.

[23] MILIOTIS, P.: "Integer Programming Approaches to the Travelling Salesman Problem", Mathematical Programming, Vol.10, pp.367-378, 1976.

[24] MILIOTIS, P.: "Using Cutting Planes to Solve the Symmetric Travelling Salesman Problem", Mathematical Programming, vol.15, pp.177-188, 1978.

[25] MILIOTIS, P., LAPORTE, G., NOBERT, Y.: "Computational Comparison of Two Methods for Finding the Shortest Complete Cycle or Circuit in a Graph", RAIRO (recherche opérationnelle), vol.15, pp.233-239, 1981.

[26] MOLE, R.H., JAMESON, S.R.: "A Sequential Route-building Algorithm Employing a Generalised Savings Criterion", Operational Research Quarterly, vo.27, pp.503-511, 1976.

[27] PIERCE, J.F.: "Direct Search Algorithms for the Truck-Dispatching Problem", Transportation Research, vol.3, pp.1-42, 1969.